JUNTA DE ANDALUCIA

CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Soporte a la Administración Electrónica

Configuración SSL para aplicaciones integradas con Plataformas de la Administración Electrónica

Sevilla, Enero de 2009
Versión 1.0

Hoja de control		
Fecha	**Autor**	**Descripción**
26/01/2009	rgf	Versión 1.0

Índice

Alcance del documento

Este documento describe la configuración SSL necesaria para aquellas aplicaciones que actualmente:

- Realicen accesos a Web Services de plataformas de la Administración Electrónica en modo seguro.

Además, aquellas aplicaciones que migren del servicio de Autenticación Web y cambien a la Fachada de Tickets de @firma o a los servicios Nativos de @firma, también deberán adecuar la configuración SSL de sus aplicativos para un correcto funcionamiento de la integración con las distintas plataformas de la Administración Electrónica.

Error Conocido

Como práctica habitual de integración con los servicios web, se introducía la configuración SSL necesaria en el método previamente a realizar la llamada, para así establecerla convenientemente. Un esquema de código sería el siguiente:

ProcesaDatosAction.java

```
/* Clase donde realizo una petición vía WS*/
// ……
/Configuración SSL
System.setProperty("javax.net.ssl.trustStore",”ruta_trustore"));
System.setProperty("javax.net.ssl.trustStorePassword","trustStorePassword");
System.setProperty("javax.net.ssl.keyStore","ruta keyStore");
System.setProperty("javax.net.ssl.keyStorePassword","keyStorePassword");
System.setProperty("java.protocol.handler.pkgs","handler");
System.setProperty("javax.net.ssl.keyStoreType", keyStoreType);
System.setProperty("javax.net.ssl.keyStoreAlias","keyStoreAlias");
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());

ObjetoRespuestaWS respuesta = ObjetoWS.llamadaMetodoWS(parametro1,parametro2);
//……
```

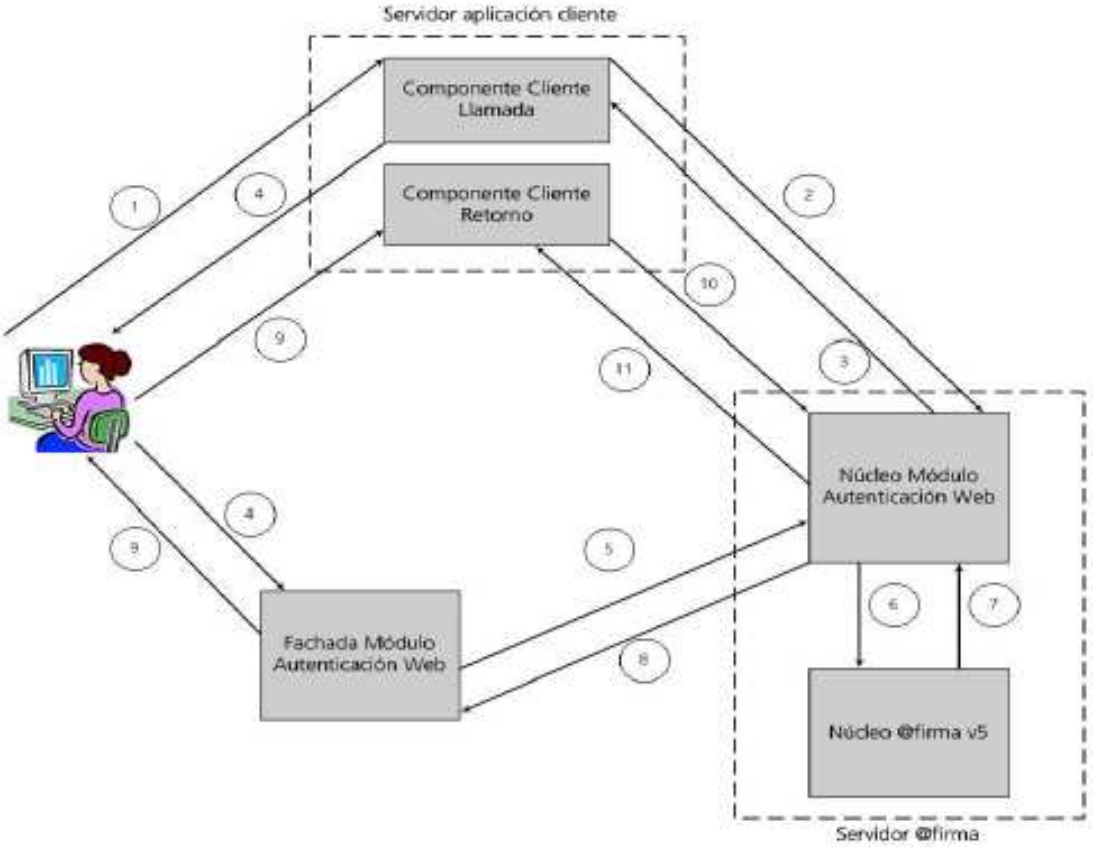
En el caso de realizar posteriormente una nueva llamada a WS, definiendo una configuración SSL diferente a la anteriormente realizada, se obtendrá el siguiente error al realizar la llamada:

```
javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure
```

Problemática de la integración

Sustitución de la Autenticación Web por Fachada de Tickets

Al sustituirse en una aplicación la Autenticación Web por la Fachada de Tickets, es necesario que se hagan llamadas a Web Services en modo seguro contra @firma siendo obligatorio realizar la comunicación por https. El proceso de autenticación por Fachada de Tickets sigue el siguiente esquema:

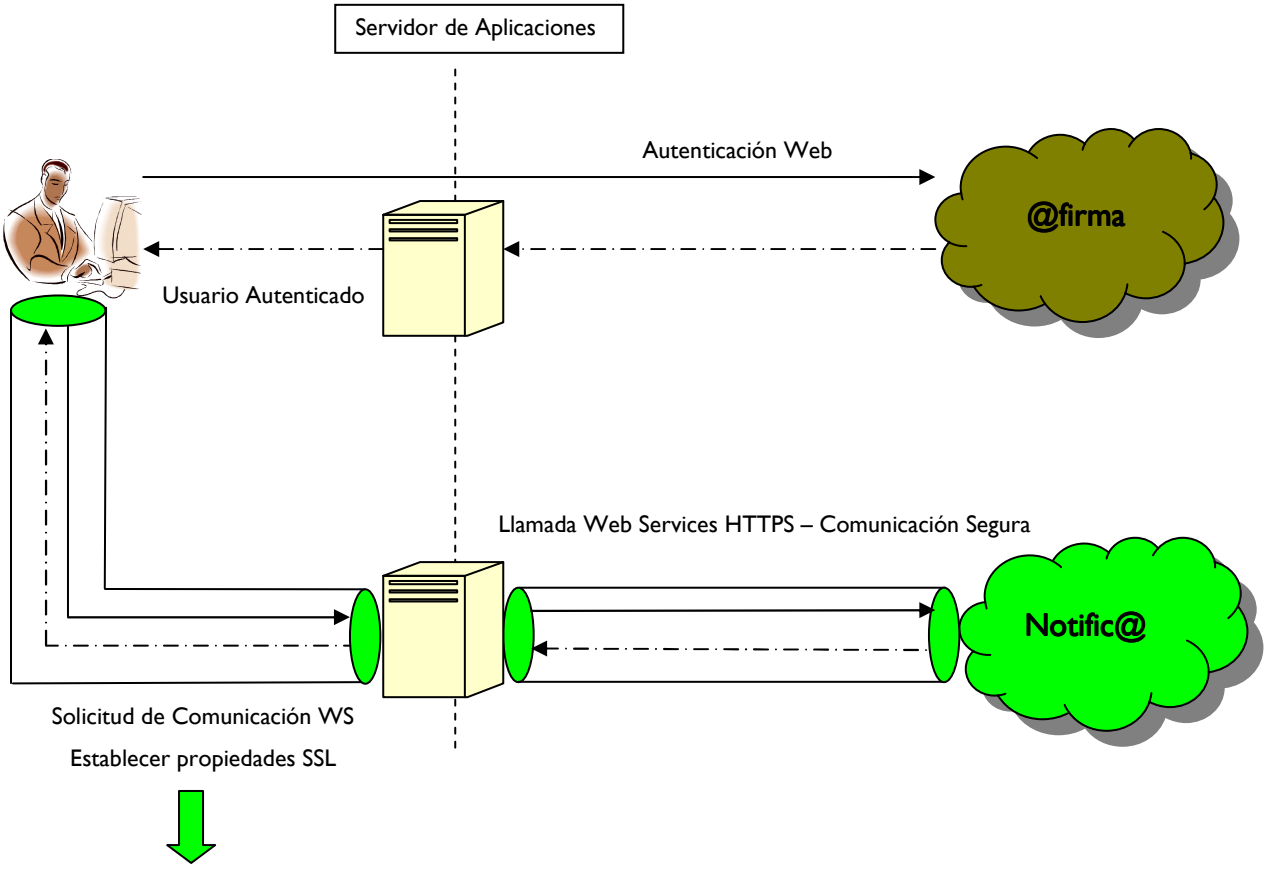


En el caso que la aplicación deba realizar otras llamadas seguras a Web Services que también hagan uso del protocolo SSL -en la Administración Electrónica las plataformas Notific@, Port@firmas o Bus Wand@ ofrecen comunicaciones seguras para el acceso a sus servicios- se recomienda que el establecimiento de las propiedades de la comunicación SSL, se realicen con anterioridad al proceso de Autenticación por Fachada de Tickets, con objeto de garantizar de durante toda la ejecución se mantengan las propiedades, evitando problemas conocidos al establecer nuevas propiedades de conexión SSL.

Situación Actual

En el caso de la Autenticación Web no es preciso definir un cacert que contenga la parte pública de los certificados existentes en @firma. A continuación se muestra un ejemplo de llamadas con el servicio de Autenticación Web junto con otro Plataforma de la Administración Electrónica.

Autenticación Web + Llamada a Web Services (Notific@)

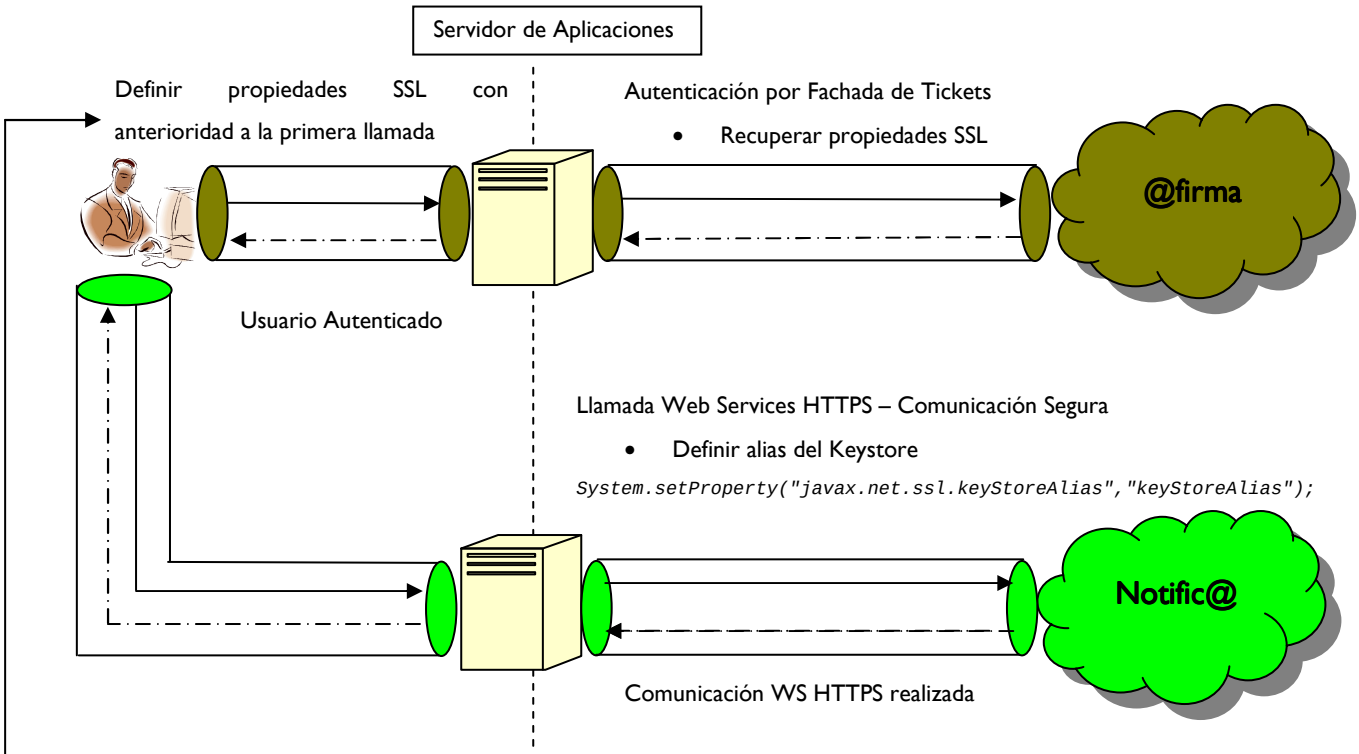



```
System.setProperty("javax.net.ssl.trustStore","ruta_trustore"));
System.setProperty("javax.net.ssl.trustStorePassword","trustStorePassword");
System.setProperty("javax.net.ssl.keyStore","ruta keyStore");
System.setProperty("javax.net.ssl.keyStorePassword","keyStorePassword");
System.setProperty("java.protocol.handler.pkgs","handler");
System.setProperty("javax.net.ssl.keyStoreType", keyStoreType);
System.setProperty("javax.net.ssl.keyStoreAlias","keyStoreAlias");
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
```

Situación tras la inclusión de la Fachada de Tickets

Con la sustitución de la Autenticación Web por la Fachada de Tickets, aquellas aplicaciones que estén integradas además con otras plataformas que hagan uso de Servicios Web, deberán configurar adecuadamente las propiedades de su comunicación SSL con anterioridad al primer acceso a Web Services que se realice desde la aplicación.

Autenticación por Fachada de Tickets + Llamada a Web Services (Notific@)




```
System.setProperty("javax.net.ssl.trustStore","ruta_trustore"));
System.setProperty("javax.net.ssl.trustStorePassword","trustStorePassword");
System.setProperty("javax.net.ssl.keyStore","ruta keyStore");
System.setProperty("javax.net.ssl.keyStorePassword","keyStorePassword");
System.setProperty("java.protocol.handler.pkgs","handler");
System.setProperty("javax.net.ssl.keyStoreType", keyStoreType);
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
```

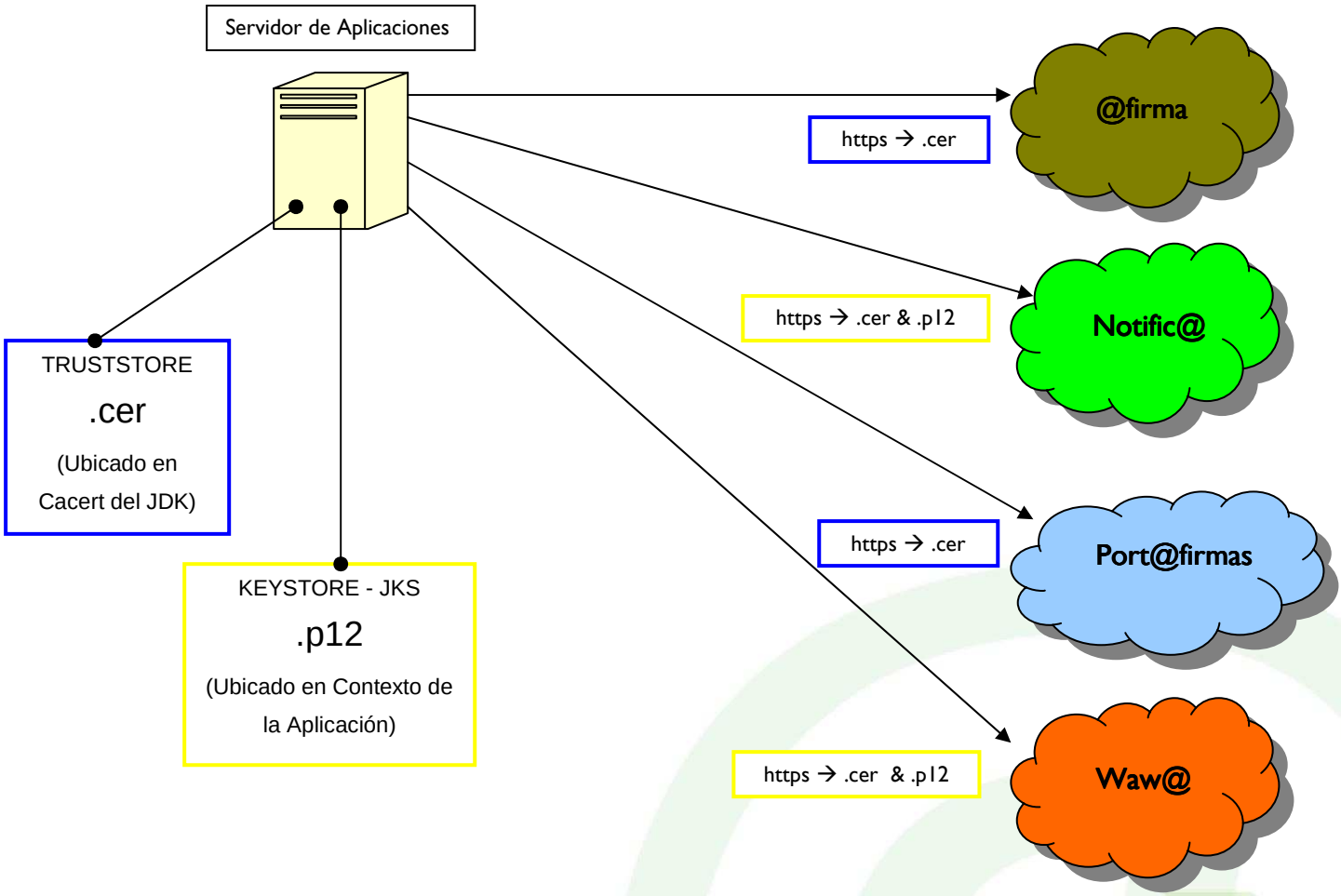
Solución para los integradores

Ante este problema para el establecimiento de las comunicaciones seguras, se recomienda la definición previa a cualquier llamada contra Web Services o a la Fachada de Tickets de @firma de las propiedades SSL oportunas. Recordar que dichos parámetros además se definirán para un correcto funcionamiento de la siguiente manera:

- TrustStore: Almacén de certificados que contenga todas las partes públicas necesarias (archivos *.cer). Se recomienda ubicarlos en el cacert del JDK que haga uso el servidor de aplicaciones, de manera que si existen varias aplicaciones desplegadas en un mismo servidor no se generen conflictos.

- KeyStore: Almacén de certificados que contenga todas los partes privadas necesarias (archivos *.p12). Se recomienda usar un almacén de tipo JKS. Se debe ubicar dentro del contexto de la aplicación. En caso de existir varios p12 en el JKS se podrán referenciar usando un alias para cada petición

Ejemplo de esquema para una aplicación integrada con @firma, Notific@, Port@firmas y Bus W@nda



TrustStore :

1. Parte pública de @firma (.cer)
2. Parte pública de Notific@ (.cer)
3. Parte pública de Port@firmas (.cer)
4. Parte pública del Bus W@nda (.cer)

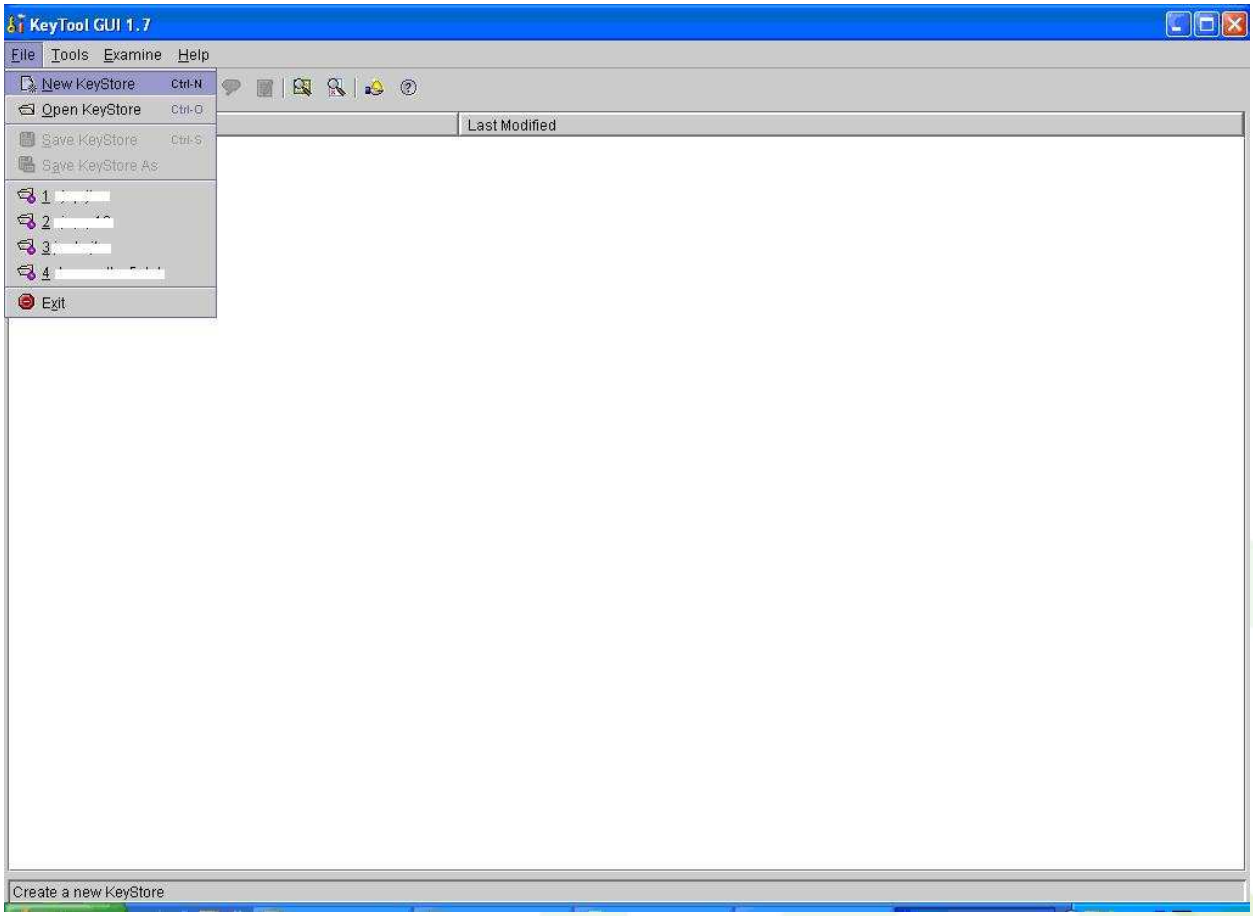
KeyStore (formato JKS):

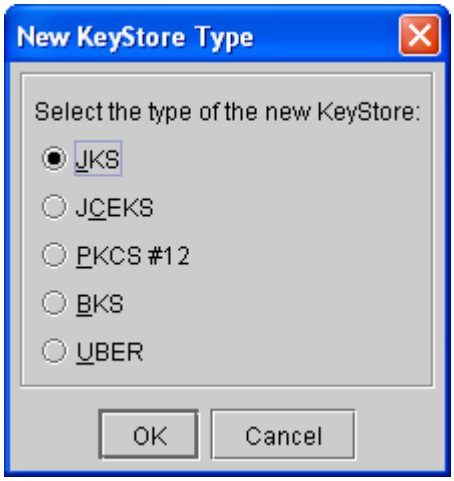
1. Parte privada de la parte pública que posee la plataforma Notific@ (.p12)
2. Parte privada de la parte pública que posee la plataforma Bus W@nda (.p12)

Ejemplo de Configuración. Código Fuente

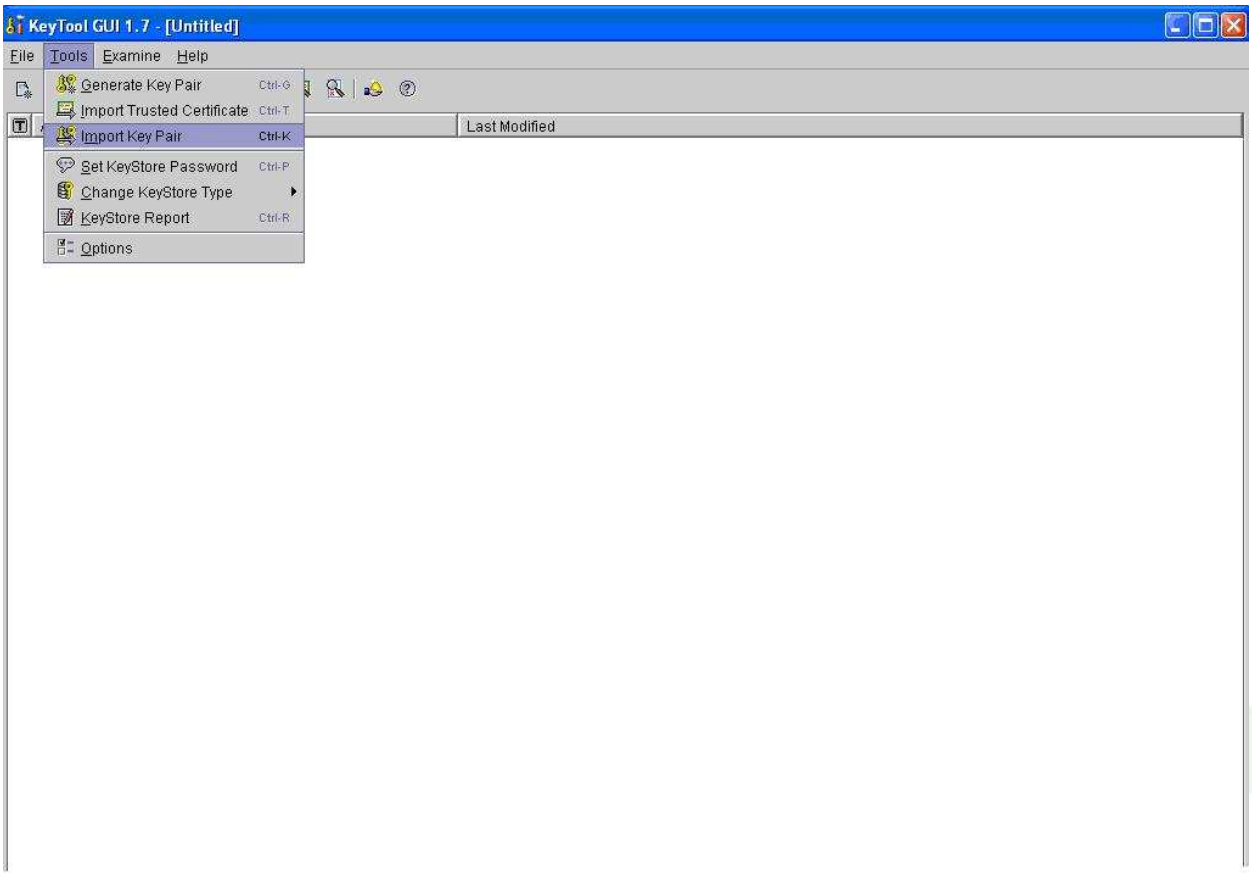
Para la generación de los almacenes de confianza se puede realizar en modo comando desde la consola del servidor, o haciendo uso de una herramienta de escritorio como KeyTool GUI. Se explica a continuación la manera de construir esos almacenes con la herramienta KeyTool.

- **Construir Almacen de Confianza KeyStores (A ubicar en el contexto de la Aplicación como JKS)**

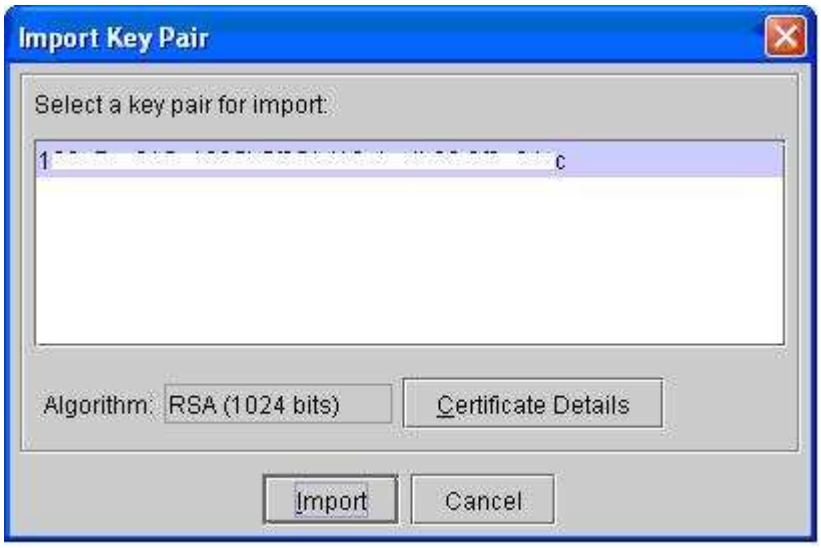




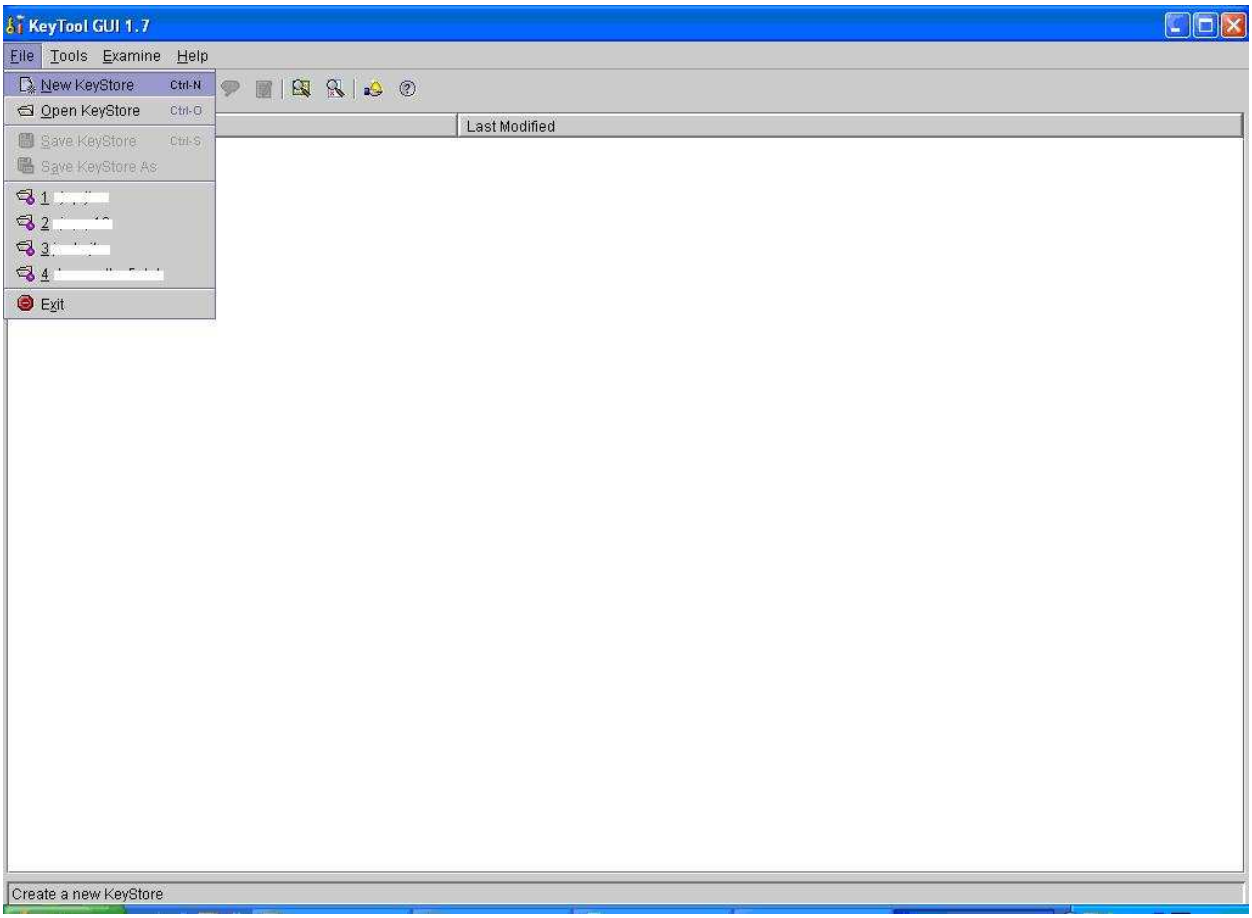
Se selecciona la opción 'Import Key Pair' facilitando el archivo *.p12. Este archivo estará protegido por contraseña, por lo que habrá que facilitarla para completar la importación.

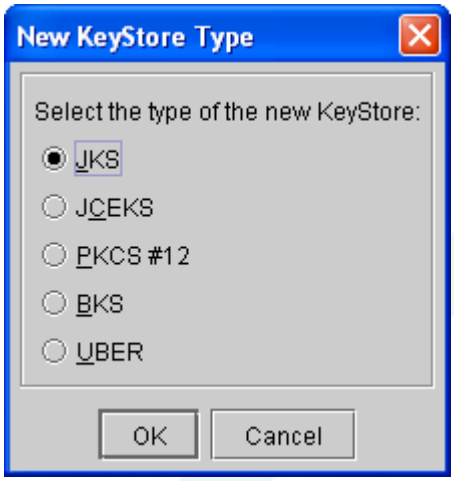


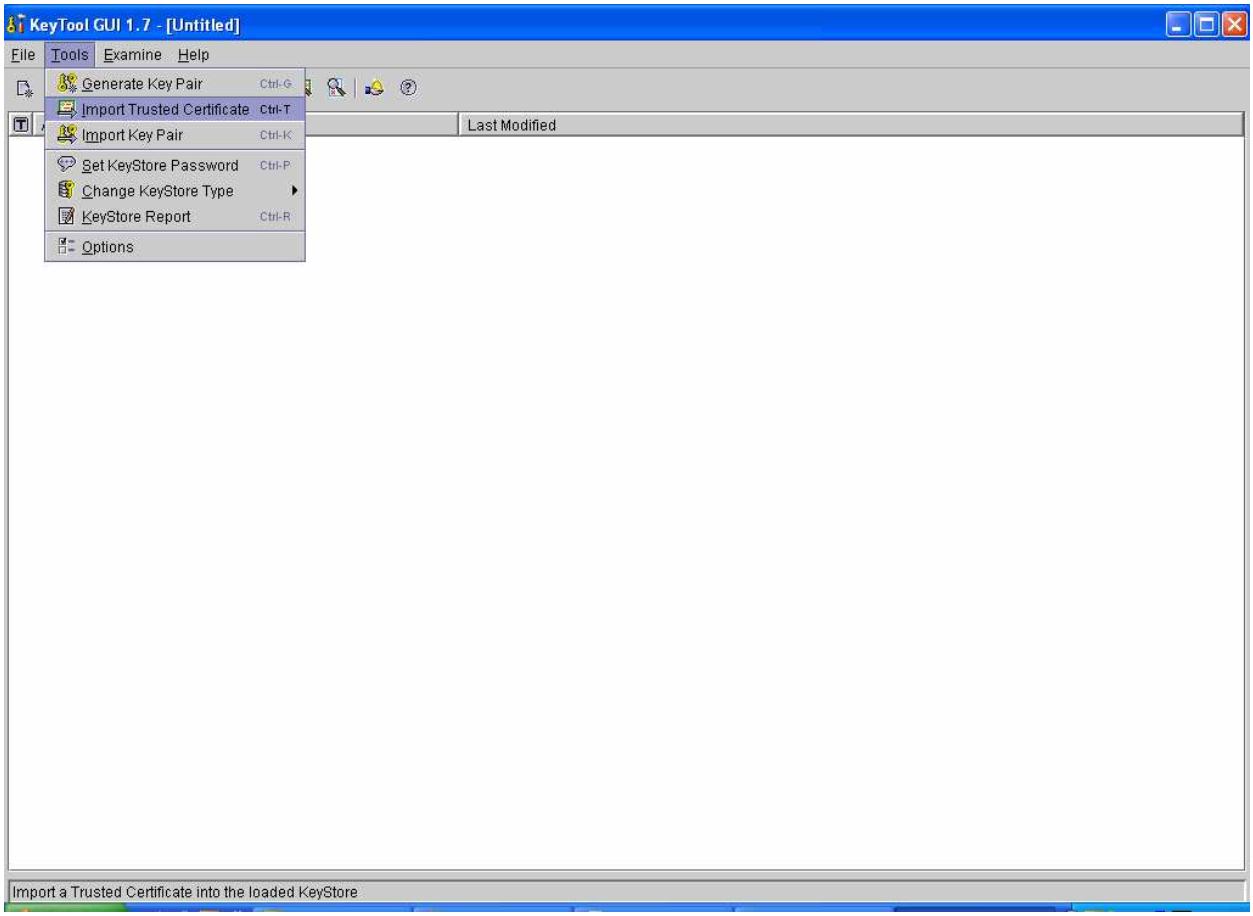
Se muestran los datos del certificado, se realiza la importación, definiendo un alias y la clave del JKS, guardando el ficheros indicando la extensión que se usará.

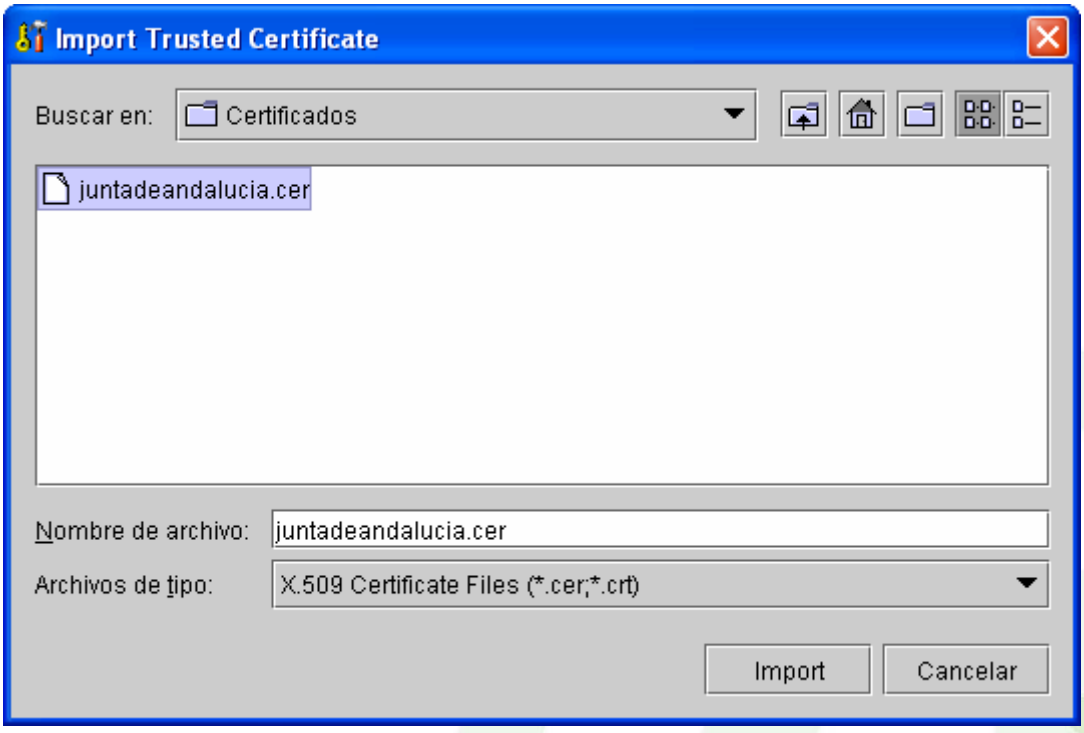


- **Construir Almacen de Confianza TrustStores (A ubicar en el Cacert del JDK)**

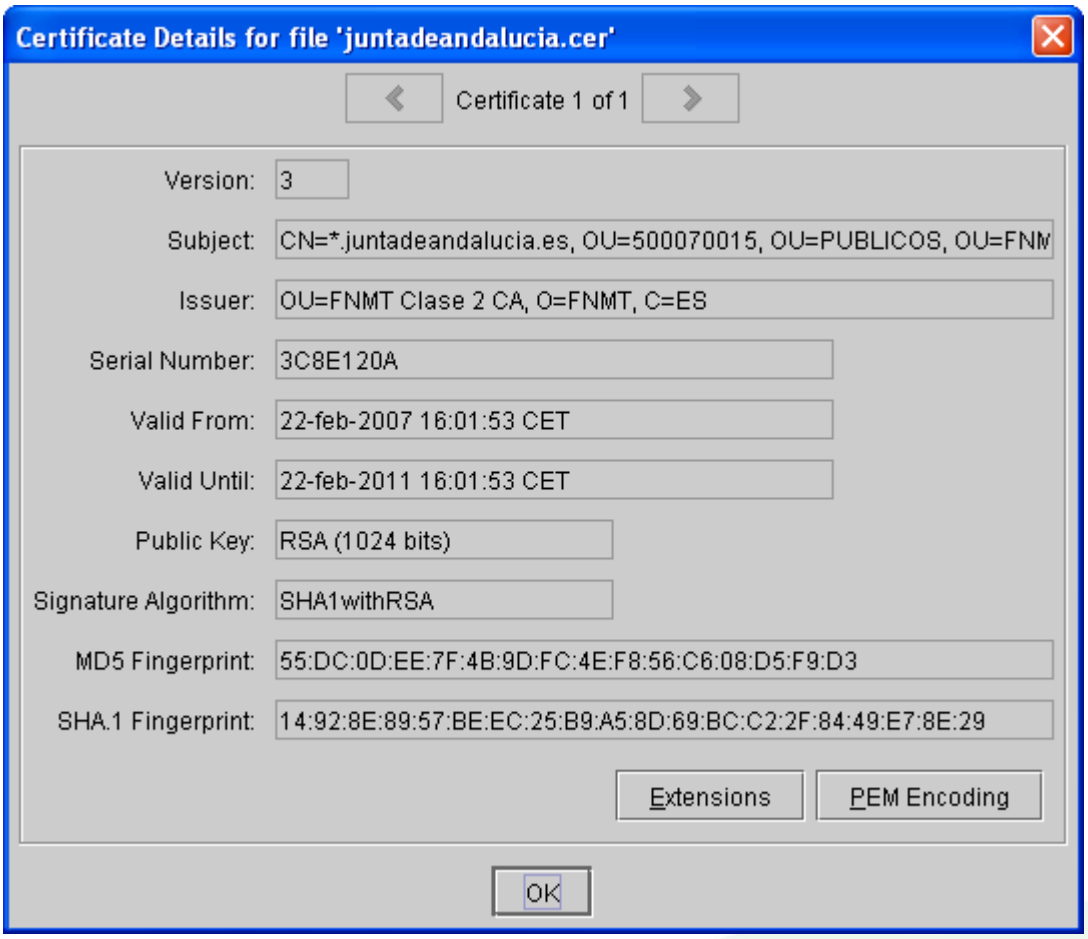




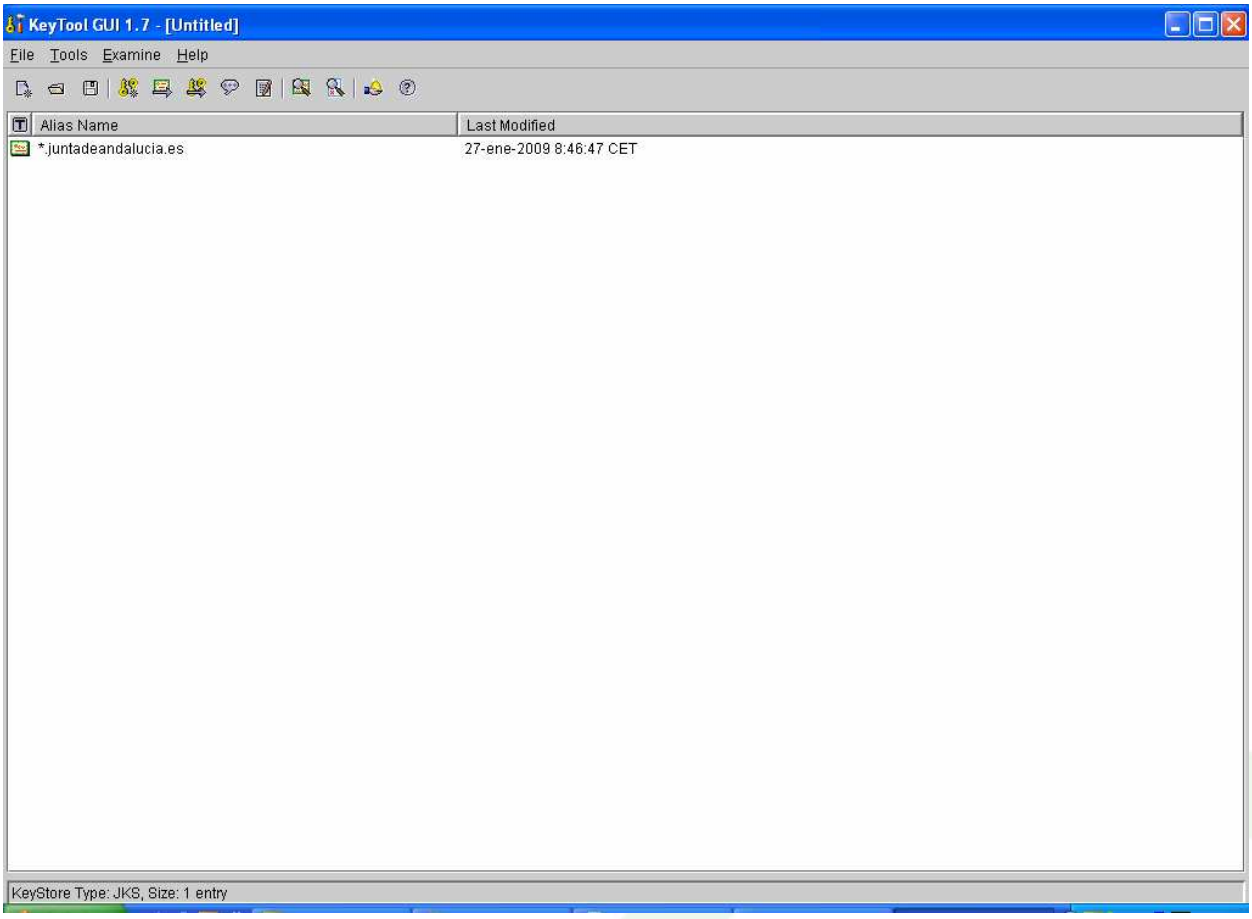

KeyTool GUI 1.7 - [Untitled]
File
Tools
Examine
Help
Generate Key Pair Ctrl-G
Import Trusted Certificate Ctrl-T
Import Key Pair Ctrl-K
Set KeyStore Password Ctrl-P
Change KeyStore Type
KeyStore Report Ctrl-R
Options
Last Modified
Import a Trusted Certificate into the loaded KeyStore


Import Trusted Certificate
Buscar en:
Certificados
juntadeandalucia.cer
Nombre de archivo:
juntadeandalucia.cer
Archivos de tipo:
X.509 Certificate Files (*.cer;*.crt)
Import
Cancelar

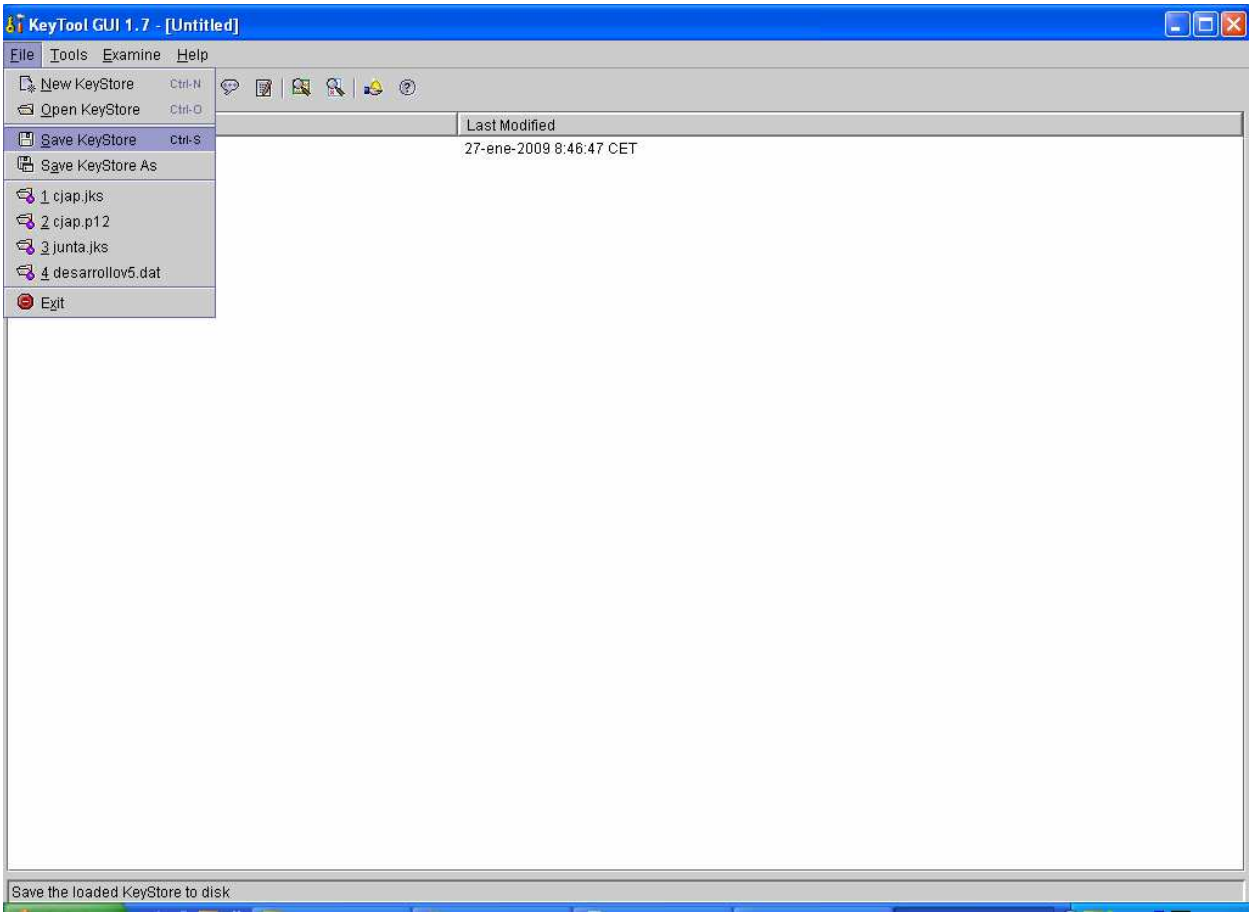

Import Trusted Certificate
Could not establish a trust path for the certficate.
The certficate information will now be displayed after which you may confirm whether or not you trust the certificate.
Aceptar

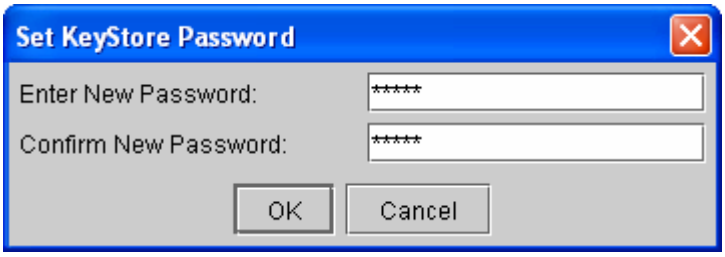

Certificate Details for file 'juntadeandalucia.cer'
Certificate 1 of 1
Version: 3
Subject: CN=*.juntadeandalucia.es, OU=500070015, OU=PUBLICOS, OU=FNM
Issuer: OU=FNMT Clase 2 CA, O=FNMT, C=ES
Serial Number: 3C8E120A
Valid From: 22-feb-2007 16:01:53 CET
Valid Until: 22-feb-2011 16:01:53 CET
Public Key: RSA (1024 bits)
Signature Algorithm: SHA1withRSA
MD5 Fingerprint: 55:DC:0D:EE:7F:4B:9D:FC:4E:F8:56:C6:08:D5:F9:D3
SHA.1 Fingerprint: 14:92:8E:89:57:BE:EC:25:B9:A5:8D:69:BC:C2:2F:84:49:E7:8E:29
Extensions
PEM Encoding
OK

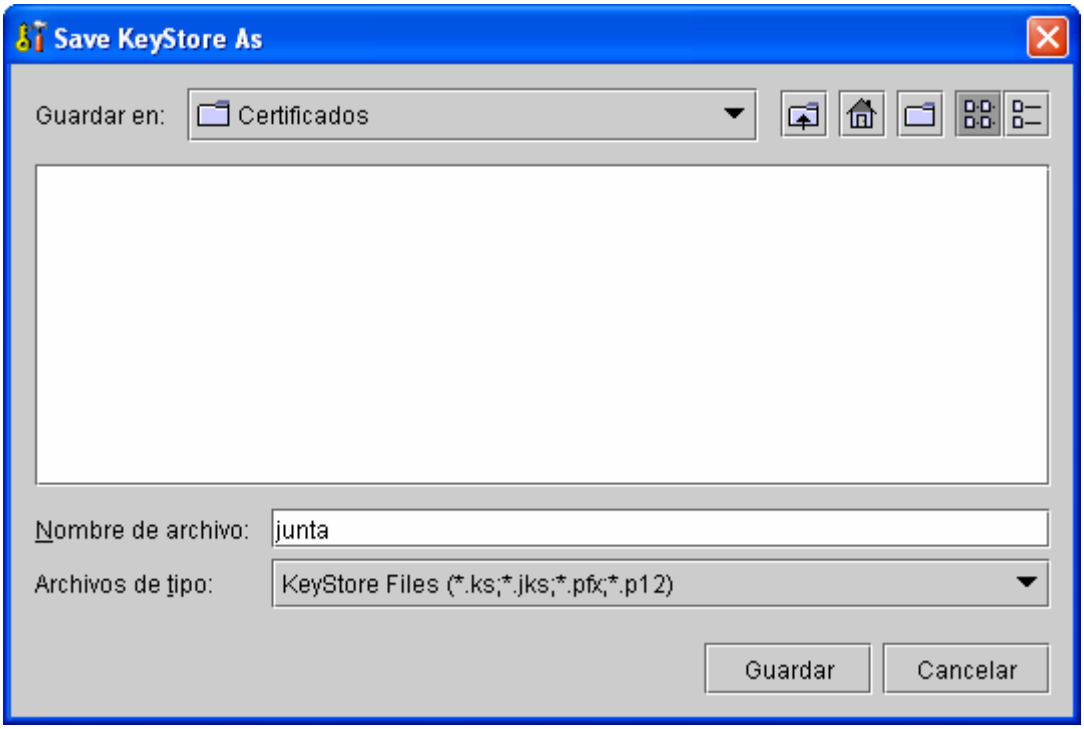

Import Trusted Certificate
Do you want to accept the certificate as trusted?
Sí
No



Repitiendo esta operación con todos los certificados necesarios se van añadiendo los diferentes certificados necesario para generar el TrustStore necesaraio para establecer la comunicación segura.


KeyTool GUI 1.7 - [Untitled]
File Tools Examine Help
New KeyStore Ctrl-N
Open KeyStore Ctrl-O
Save KeyStore Ctrl-S
Save KeyStore As
1 cjap.jks
2 cjap.p12
3 junta.jks
4 desarrollov5.dat
Exit
Last Modified
27-ene-2009 8:46:47 CET
Save the loaded KeyStore to disk


Set KeyStore Password
Enter New Password:
*****
Confirm New Password:
*****
OK
Cancel



Una vez añadido todos los certificados, procedemos a guardar el Almacén. En la integración con @firma se facilita este archivo sin extensión o con una extensión .dat, normalmente dispuesto en un almacén con clave 'changeit', que en los entornos de desarrollo local del integrador puede ubicarse en cualquier parte del contexto de su aplicación con esa misma extensión.

Para entornos de preproducción o explotación, los certificados deben añadirse al cacert del jdk que esté usando el servidor de aplicaciones. En esos casos, el uso de la herramienta KeyTool podría no ser posible, por lo que para añadir los certificados al cacert del JDK debe realizarse en modo comandos:

```
> keytool -import -alias wildcardJunta -file wildcardJunta.cer -keystore
$rutaAlKeystoreDeLaAplicacion/nombreAlmacenDeCertificados
```

- **Definición de las propiedades SSL en el código fuente**

Tras construir los almacenes y ubicarlos convenientemente con todos los partes publicas (.cer) y pares de claves (.p12), es necesario referenciar en una parte del código previa a la 1° invocación las propiedades de la comunicación segura.

```
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());

//Posible valor: com.sun.net.ssl.internal.www.protocol.https
System.setProperty("java.protocol.handler.pkgs","handler");
//Posible valor: JKS o PKCS12
System.setProperty("javax.net.ssl.keyStoreType", "keyStoreType");

// Las rutas en servidores Linux /opt/jdk1.5/jre/lib/security/cacerts
// Las rutas en servidores Windows separadas por \\
C:\\Desarrollo\\myapp\\WebContent\\WEB-INF\\classes\\desarrollov5.dat
System.setProperty("javax.net.ssl.trustStore","ruta_trustStore");
System.setProperty("javax.net.ssl.trustStorePassword","trustStorePassword");
System.setProperty("javax.net.ssl.keyStore","ruta_keyStore");
System.setProperty("javax.net.ssl.keyStorePassword","keyStorePassword");
System.setProperty("javax.net.ssl.keyStoreAlias","keyStoreAlias");
```

El significado de las diferentes variables es:

- trustStore, ruta donde se encuentra el almacén de confianza (conjunto de *.cer)
- trustStorePassword, contraseña del almacén de confianza
- trustStoreType, tipo de almacén, en este caso JKS
- keyStore, ruta con el almacén de claves donde se encuentra el certificado (par de claves, normalmente en formato p12)
- keyStorePassword, contraseña del almacén de claves
- keyStoreType, al igual que con el almacén de confianza aquí indicaremos el tipo de almacén
- keyStoreAlias, en el caso de tener más de un certificado en el almacén de claves se tendrá que indicar cual de ellos es el de la aplicación mediante el alias que tiene definido